Realistic Driving Trips For Location Privacy

John Krumm

Microsoft Research
Microsoft Corporation
One Microsoft Way
Redmond, WA 98052 USA
jckrumm@microsoft.com

Abstract. Simulated, false location reports can be an effective way to confuse a privacy attacker. When a mobile user must transmit his or her location to a central server, these location reports can be accompanied by false reports that, ideally, cannot be distinguished from the true one. The realism of the false reports is important, because otherwise an attacker could filter out all but the real data. Using our database of GPS tracks from over 250 volunteer drivers, we developed probabilistic models of driving behavior and applied the models to create realistic driving trips. The simulations model realistic start and end points, slightly non-optimal routes, realistic driving speeds, and spatially varying GPS noise.

Keywords: location privacy, location-based services, false trips, GPS

1 Trip Simulations For Privacy

Some location-based services require users to transmit location from their mobile device to a central server. These transmissions can be user-initiated and sporadic, such as a query to find nearby restaurants. Other location transmissions can be periodic and relatively frequent, like those querying for alerts about nearby friends, events, and advertising. These location transmissions and the responses from the server could be compromised by an attacker, resulting in a potentially sensitive privacy leak.

One approach to bolstering privacy is to anonymize the location transmissions by stripping away any identifying information. The server often still requires a pseudonym, however, in order to know how to respond and to whom. It has been shown in [6] that an attacker can find a person’s home even with pseudonomized GPS tracks, and [10] shows how such an attack can go further and find the actual name of the victim based on publicly available street address listings. Even using completely anonymized tracks, with no pseudonym, [4] has shown how to find which location points belong together in the same track, effectively creating a pseudonym for each trip.

Another commonly proposed technique for improving location privacy is obfuscation. This approach degrades the transmitted location in some way that reduces the chance that an attacker can find the potential victim’s true location.

Obfuscation techniques include inaccuracy and imprecision, introduced for location privacy in [1]. Inaccuracy can be achieved by adding random noise to location measurements, and imprecision can be achieved by snapping measurements to a grid. Unfortunately, [10] showed that the amount of obfuscation necessary to foil an attack can be very high, *e.g.* an identity attack still worked after adding noise with a 1-kilometer standard deviation. Gruteser and Grunwald [3] introduced *k*-anonymity for location privacy, in which point location reports are replaced by regions containing k-1 other people, another way of achieving imprecision. While obfuscation can be effective, it necessitates the degradation of the location data, which can be fatal for certain applications.

One little-explored but promising technique for location privacy is for the user to send several false location reports along with the real one. The server would respond to all the reports, and the user would ignore all but the response to their actual location. With enough false reports, the chances of an attacker picking the true one could be reduced to an acceptable level. This technique uses no obfuscation, meaning it would still work for location-based services that require accurate and precise point reports, such as alerts of nearby friends and location-based advertising. The only previous work exploring this idea appears to be that of Kido *et al.* [9] who explore an algorithm for reducing the inevitable increase in communication cost.

The effectiveness of false reports depends heavily on minimizing the ability of an attacker to determine which reports are false. Reporting completely random locations is risky, because they may fall at obviously unlikely locations like lakes, oceans, swamps, and rugged mountains. Furthermore, since locations from the true report will follow a plausible path, the false reports must also be plausible paths. Otherwise, the continuity of the true path would be easy to distinguish from the “twinkling” of the false reports.

The Kido paper concentrates on reducing communication costs, so its two proposed false path generation techniques are not emphasized. One of these techniques, “Moving in a Neighborhood”, is essentially a random walk model, while the other, “Moving in a Limited Neighborhood”, modifies the first to avoid clumping false reports near other users’ true locations. However, Duckam *et al.* [6] point out sophisticated techniques that can be used to filter out false reports. For instance, they note that movement may be constrained to a graph, like a road network. Also, people normally move with a goal in mind. Thus, random walk models are likely to be easily identifiable by an attacker who could then strip away all but the true location report.

Related to our work is research on mobility patterns to model the use of wireless networks. For mobile networking, mobility simulations are important for wireless networking with both fixed base stations [11] and mobile peers [1, 3]. Because fixed base stations normally have a large range, the associated mobility models can work at the relatively coarse level of cells surrounding each base station, as in [18]. For mobile ad hoc networks (MANETS), however, finer grained simulations are necessary due to the short range of the participants’ radios. Such models are used to help simulate a collection of wireless nodes, such as automobiles, forming a network with no central control. The Random Waypoint model [2] is one of the first simulations relevant to this situation. Here, a subject moves in a straight line toward a randomly chosen waypoint at a randomly chosen speed, then chooses another waypoint and speed, *etc*. Other such mathematical models have been developed since, all aimed at increasing realism. For the case where the mobile nodes are vehicles [8], as in this paper, one of the more sophisticated models constrains the vehicles to a road network, either random or from a real map [20]. These mathematical models fall short of reality, however, because they lack the degrees of freedom to faithfully simulate real drivers. Maximum realism comes from trace-based models that use actual path traces played back from real subjects. These are limited, however, because measuring traces is relatively expensive, especially for high volumes of traffic in cities.

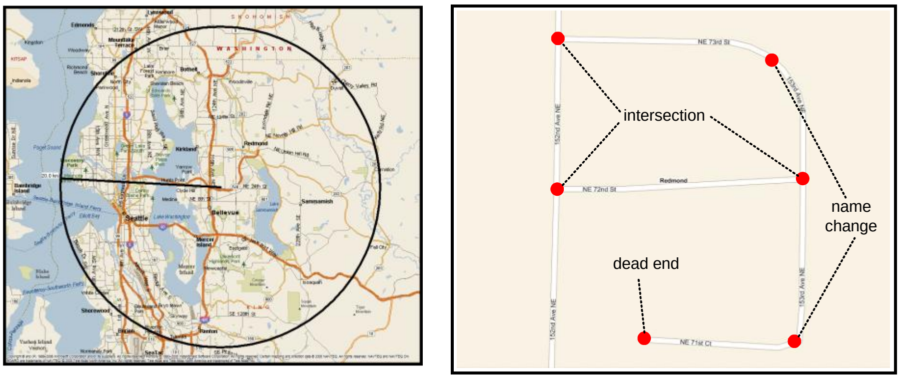
We also note that simulated trips for privacy *vs*. wireless networking have different goals, and therefore different criteria. For instance, mobility simulations for wireless networking often try to account for group behavior and interactions among mobile nodes, because this can affect loads on base stations and present opportunities for messages to hop between peers. For privacy, however, our goal is to fool an attacker, which means we can give many isolated, false trips that do not need to show any regard for each other.

This paper presents simulated traces based on an actual road network. Our method approaches the realism of actual traces by using probabilistic models of driving behavior abstracted from real traces. Our simulated driving trips exhibit these realistic characteristics, all derived from a statistical analysis of actual driving traces:

- Realistic starting and ending points
- Goal-directed routes with randomness
- Random driving speeds
- Spatially varying GPS noise

We can generate an arbitrary number of these traces, all of which adhere to the statistical behaviors we see for actual drivers.

The following sections describe how we model each of these characteristics, preceded by a description of our measured driving data.



(a) (b)

Figure 1: (a) We analyzed and generated trips inside the 20 kilometer radius circle covering area around Seattle, Washington, USA. (b) Our road network is anchored by nodes that occur at intersections, dead ends, and road name changes.

2 Multiperson Location Survey

Our statistical behavior models are based on observations of where drivers drive measured from GPS receivers. We have been gathering GPS data from volunteer drivers in our Multiperson Location Survey (MLS) starting in March of 2004. Volunteer drivers are loaned one of our 55 Garmin Geko 201 GPS receivers, capable of recording 10,000 time-stamped latitude/longitude measurements. The GPS receivers are set to an adaptive recording mode that records more points when the vehicle is moving and accelerating. The median interval between recorded points is 6 seconds and 62 meters.

For this study, we used data from 253 subjects. From these subjects, we have approximately 2.3 million time-stamped latitude/longitude points comprising about 16,000 separate trips. We split the sequence of points into individual trips at gaps of more than five minutes and at apparent speeds of more than 100 miles per hour. We also eliminate trips with fewer than 10 measured points. High apparent speeds and unusually short trips often come from random, noise-induced measurements while a vehicle is parked.

Approximately 80% of our GPS data is contained in a 20 kilometer radius circle centered in the Seattle, Washington, USA region, so we limited our analysis to this area, shown in Figure 1(a).

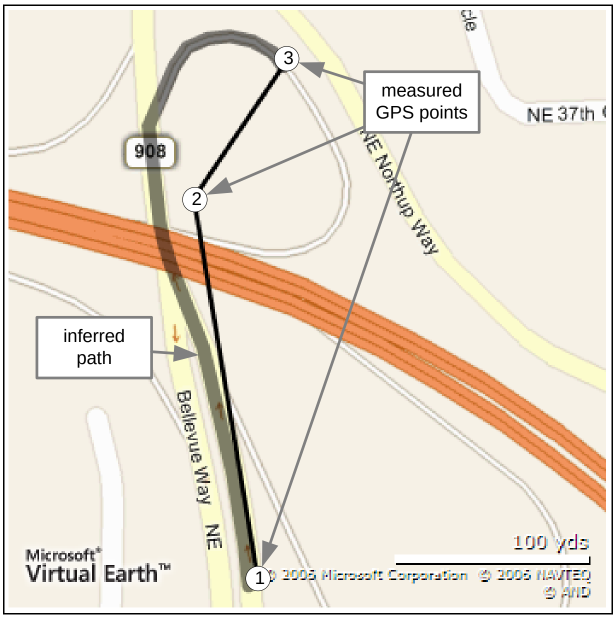


Figure 2: We used a map-matching algorithm to determine which roads correspond to noisy GPS points. From [11].

3 Simulating Trip Endpoints

The first step in our simulation is choosing start and end points of a trip. Vehicle trips normally start and end near a road, and some parts of a geographic region are more popular than others. We attempt to model this behavior, first, by constraining starting and ending points to nodes in a road network. The road network is a graph, in a mathematical sense, where roads are edges and nodes occur at intersections, dead ends, and changes in the road name, as shown in Figure 1(b). Our analysis region (Figure 1(a)) contains 51,637 nodes and 65,549 edges with an average length of 131 meters. While actual trips could start or end almost anywhere, our nodes give a convenient spatial sampling of the geographic space. An attacker may notice that the false trips start only on nodes, but this is mitigated somewhat by the random GPS noise we add, described in Section 7.

Our goal is to compute a probability for each node governing the chances that a trip will start or end there. Toward this end, we first examine our GPS data to find the node nearest to the start and end of each actual trip. In subsequent sections, we need to know the *entire* sequence of nodes for each trip, which we compute with a probabilistic map-matching technique [11], illustrated in Figure 2. This algorithm takes as input a sequence of time-stamped latitude/longitude points and produces a sequence of nodes that best represents the trip. The map-matching algorithm uses a hidden Markov model to produce a route that simultaneously minimizes the GPS error and accounts for the GPS time stamps in light of the road network's connectivity and speed limits. After processing each GPS trip, we have a time-stamped sequence of nodes and edges for each one, including the start and end nodes.

We examined a variety of features of the nodes to compute the probability $P(n_i)$ that a node n_i will be a start or end point of a trip. The features are shown in Table 1. All except the "USGS" (United States Geological Survey) and "Roads attached" features are actually features of road edges, not nodes. To compute the corresponding node feature, we let the attached edges vote for the feature value and take the

Table 1: These are the features that determine the probability of a node being chosen as an endpoint of a trip.

Feature	Values	"true" probability
Autos allowed	true/false	1.000
Ferry route	true/false	0.000
Paved road	true/false	0.997
Private road	true/false	0.050
Roundabout	true/false	0.001
Through traffic	true/false	0.965
Toll	true/false	0.000
USGS	21 ground types	--
Roads attached	1,2,3,4,5,6	--
Number of lanes	1,2,4	--
Road type	7 road types	--

plurality. For instance, one of the features is called "Autos allowed". This will be true if most of the node's connected roads allow cars to drive on them. The meaning of the binary features is obvious from their names in Table 1. For these features, Table 1 also gives the fraction of the endpoint nodes whose corresponding feature value was "true". For instance, of all the endpoints extracted from the GPS data, a fraction of 0.997 of them were on nodes whose plurality of attached edges was paved. Similarly, no routes started or ended on nodes whose plurality of attached edges were toll roads or ferry routes, which makes intuitive sense.

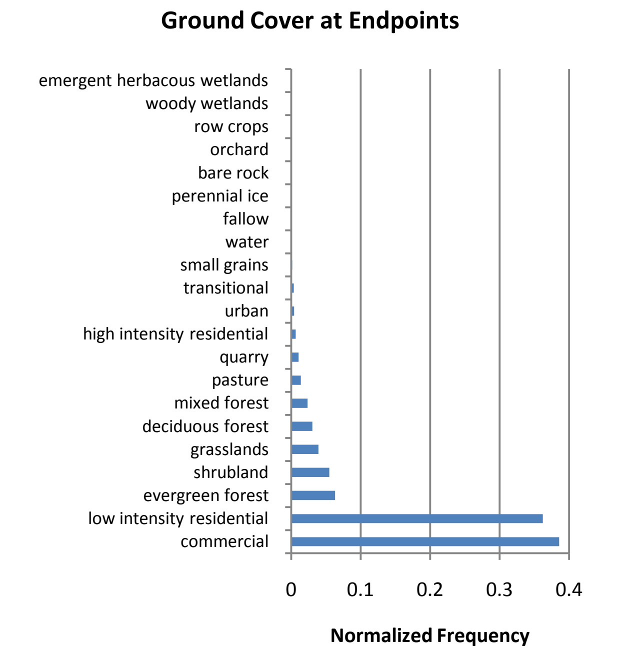


Figure 3: The relative popularity of trip endpoints varies depending on the ground cover.

The "USGS" feature pertains to the ground cover at the node, *e.g.* urban, grasslands, *etc*. The USGS makes available free, digital maps of the U.S. giving a ground cover type for each 30m x 30m square of ground [7]. The 21 ground cover types and the associated probability of an endpoint node landing on them are shown in Figure 3.

The "Roads attached" feature counts the number of roads attached to the node. The number of roads attached and associated probabilities of endpoint nodes occurring there are 1 (0.010), 2 (0.152), 3 (0.436), 4 (0.295), 5 (0.014), 6 (0.001).

"Number of lanes" is the plurality of the number of road lanes on the node's connected edges. The number of lanes and probabilities are 1 (0.751), 2 (0.241), 4 (0.009). End points most often occur on single- and double-lane roads.

"Road type" gives the plurality vote of the type of road connected to the node. The probabilities, shown in Figure 4, indicate that highways, ferries, and ramps are unpopular places to start or end a trip.

To compute the probability of a given node being an endpoint, we use a naïve Bayes formulation for the 11 features f_j from Table 1 that says

$$P_{endpoint}\ (n_i|f_1, f_2, \dots, f_{11}) = \prod_{j=1}^{11} P_{endpoint}\ (n_i|f_j) \qquad (1)$$

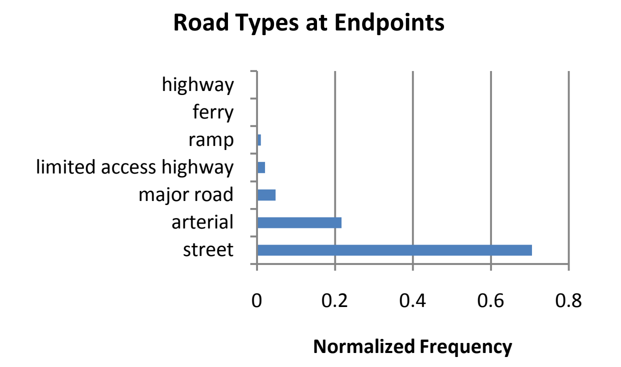


Figure 4: Nodes with a plurality of highway, ferry, or ramp connections are unpopular places to start or end a trip.

We take the $P_{endpoint}\,(n_i|f_j)$ values from the feature probabilities described above. Using naïve Bayes carries a risk of overweighting some features that have correlations with each other, but it has been shown to work well in practice [13]. With this technique, we find that the least popular endpoints are grouped along highways and also appear at the ends of unpaved or private roads.

The preceding analysis does not distinguish between the start and end point of a trip, based on the observation that each point normally serves both roles for a typical driver. However, having chosen a random *starting* point based on the probabilities in Equation (1), the *ending* point should not be chosen at an arbitrary distance away. Intuitively, we know that most car trips are measured in minutes, not hours, which limits the range of likely destinations. To quantify this intuition, we used data from the U.S. 2001 National Household Transportation Survey (NHTS) [8]. The NHTS collected data on daily and longer-distance travel from approximately 66,000 U.S. households based on travel diaries kept by participants. A histogram of trip times from this study is shown in Figure 5.

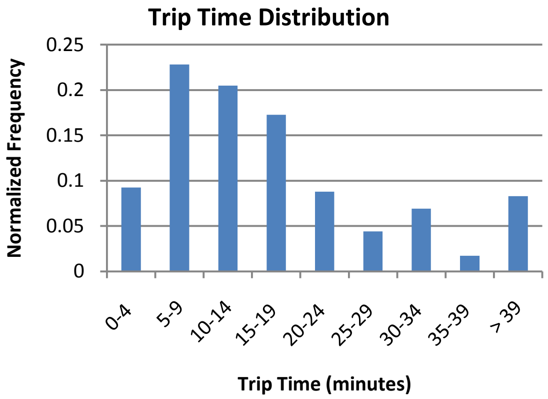


Figure 5: This distribution of trip times is used to pick random trip destinations.

We designate this distribution as $P_{trip\ time}\,(t)$, where t is the trip time. Having chosen a random starting point n_{start} from Equation (1), we compute the driving times to all the other nodes, designated as $t(n_{start}\,, n_i)$., using a conventional path planner. The probability of picking a destination node n_i is then

$$\begin{aligned} &P_{destination}\ (n_i|f_1, f_2, \ldots, f_{11}, n_{start}) \\ &= P_{endpoint}\ (n_i|f_1, f_2, \ldots, f_{11}) P(n_i|n_{start}) \\ &= P_{endpoint}\ (n_i|f_1, f_2, \ldots, f_{11}) P_{trip\ time}\left(t(n_{start}, n_i)\right) \end{aligned} \qquad (2)$$

This is simply the endpoint probability of the candidate destination node multiplied by the distribution governing trip times, evaluated at the time it would take to drive to the candidate destination. This gives the false trips the same distribution of trip times as the NHTS study suggests.

We use Equations (1) and (2) to randomly chose a start and end point of the trip, respectively. We note that the driving times used for computing $t(n_{start}, n_i)$ are based on the speed limits in the road network database, which may or may not be realistic driving speeds. For choosing a route and ultimately making a time-stamped, simulated trip, we need probability distributions governing the speeds that drivers actually drive. This is the topic of the next section.

4 Speeds At Nodes

For simulating routes and eventually time-stamped location traces, we compute probability distributions of actual driving speeds at every node from our measured GPS data. For each trip from our GPS loggers, the map-matching algorithm generates a sequence of time-stamped locations along the road network. From this, we compute a sequence of time-stamped distances along the trip, (t_i, x_i), $i = 1 \ldots N$. Here there are N points on the trip measured at times t_i. The variable x_i represents the accumulated distance along the trip, with $x_1 = 0$, x_N as the total length of the trip, and x_i monotonically non-decreasing with i. We note that the (t_i, x_i) representation is different from the more obvious, and ultimately less convenient, choice of representing our recorded trips as time-stamped latitude/longitude pairs.

Since we need to sample locations at an arbitrary interval, we interpolate (t_i, x_i) with a one-dimensional cubic spline, which gives $x(t)$ for any $t \in [t_1, t_N]$. A conventional cubic spline is not necessarily monotonic, thus the resulting wiggles in the spline could have the accumulated distance occasionally decreasing with time. We chose the monotonic cubic spline presented by Steffen [15], which is simple to implement and ensures monotonicity with time. Speed along the measured trip is simply $\dot{x}(t)$.

While $\dot{x}(t)$ approximates the speed on the trip at any point in time t, we still do not know which values of t correspond to the nodes in the road network along the driver's route. We need speed samples at these points in order to compute speed distributions at all the nodes. We solve this by computing the accumulated distance along the trip to each node encountered. From this, we can compute which particular spline section pertains to that part of the trip and then find t at that point by solving a cubic equation. Thus, each measured trip gives a sample of the drivers' speeds at each node along the way.

Using a time and distance representation (*i.e.* (t_i, x_i) as above) proved to be a good alternative over using time-stamp coordinates like $(t_i, \text{lat}_i, \text{long}_i)$. The time and distance representation made it relatively easy to interpolate points along the route without the worry of the interpolant wiggling off the road. It also made it easy to compute speeds with a simple derivative and, in Section 6, to solve a differential equation for filling in simulated locations between nodes.

With sampled speeds at each node, we can compute a histogram of speeds for each node that was encountered in actual driving by our GPS subjects. However, we want speed distributions for all the nodes in our region of study, not just the ones we measured. Toward this end, we abstract away the particular node, replace it with node features, and compute a speed histogram as a function of the feature values. The features we choose for each node are the seven possible road classifications (listed in Figure 4) and the seven possible speed limits of the approach and departure edges. With these features, we can abstract speed distributions from particular, measured nodes into all the nodes in our region of study. These features are intentionally sensitive to the characteristics of the edges used to approach and depart from the node, because we expect speeds to be sensitive to the context surrounding the node. Therefore, the same node could have multiple speed distributions depending on the roads connected to it. An example speed distribution is shown in Figure 6, which shows the speed distribution on a node that connects a limited access highway to an off ramp.

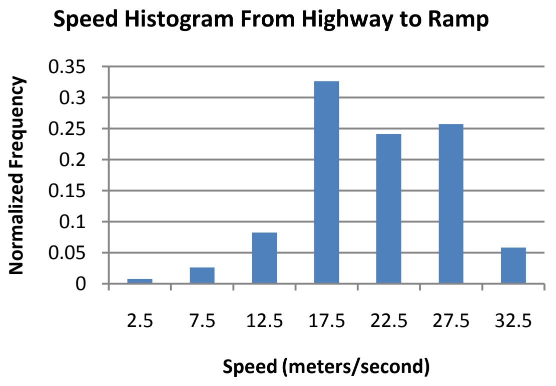


Figure 6: Speed distribution observed going from a limited access highway (speed limit approximately 26 meters/second) to a ramp (speed limit approximately 11 meters/second). The average value is 21.4 meters/second, showing that drivers are generally slowing down from the highway's maximum speed limit. This is based on 377 observations at intersections of this type.

With a four-dimensional feature vector (approach road classification, approach speed limit, departure road classification, departure speed limit), and seven possible values for each dimension, there are 7^4=2401 possible features vectors. We observed only 434 (18%) in our GPS data. We explain in the subsequent sections how we actually generated random speeds for a node depending on the intended purpose. For each feature vector, we had an average of 2257 observations from our GPS data.

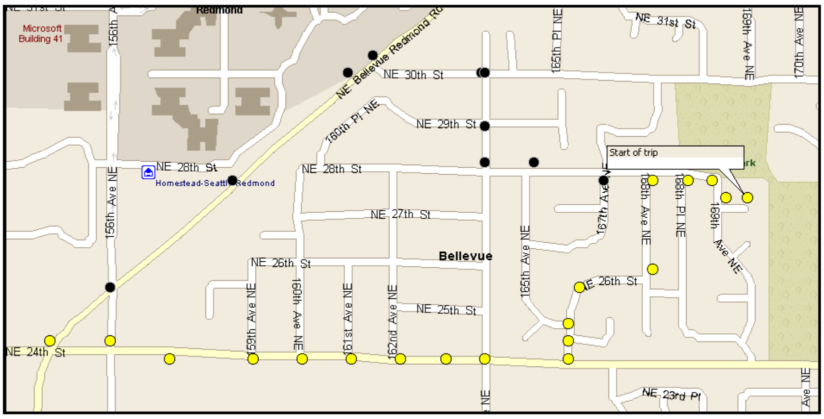


Figure 7: A trip starts at the right of the figure and moves toward the lower left. The black dots show nodes along the standard path planned using the roads' speed limits. The yellow (lighter) dots show a path planned with random speed limits.

5 Random Routes

Given random start and end points from Section 3, we could use a conventional path planner to find a reasonable, simulated route between them. However, we know from previous research that drivers do not always take the optimal route from the route planner's point of view [12]. Drivers may be unaware of the "optimal" route, they may know a better route, or they may have preferences that go beyond the route-planner's idea of optimal, *i.e.* minimum time. We want our simulated trips to appear realistic to a privacy attacker. Thus they cannot always be optimal from a route planner's point of view, because that optimality could be easily detected, even for partial trips. (A section of a minimum cost route is still the minimum cost route between the section's start and end points.)

We inject randomness into our routes by injecting randomness into the cost of all the edges. We do this by computing random speeds for the road edges from the speed distributions described in the previous section. Specifically, for each node, we draw randomly generated speeds from the speed distributions using all the possible approach/depart pairs for roads connected to the node. If we have not observed a particular approach/depart pair, we skip it. The speed assigned to the road between a pair of nodes is the average of the random speeds drawn for each of the two nodes. We generate new, random speeds before planning each route, which helps to differentiate different trips between the same start and end points. With these random speeds, we apply a standard A* search algorithm to find the minimum time route.

Figure 7 shows parts of two routes, one generated with the road network's built in speed limits and the other with random speeds as described above. Both appear reasonable.

6 Points along Route

The routes from the previous section demonstrate start points, end points, and routes that are reasonable but random. The next element is the time stamps and locations of points along the route. We want to simulate a GPS taking measurements at any frequency along the route. One simple alternative would be to take speed limits from the original road network representation and apply them to get distance along the route as a function of time. However, drivers do not drive at constant speeds along edges, they do not undergo step changes in speeds at changes in speed limits, and their behavior varies over time.

We use our random speed distributions again to generate random speeds at each node encountered on the random route. For each node, we know the characteristics of the approach and departure edges, so we use the applicable speed distribution if we have it. If not, the computed speed for the node is the average of the nominal speeds limits on the approach and departure edges. This gives a speed at each node, and we do a linear interpolation of speed between nodes, resulting in a specification giving speed as a function of distance along the route. For example, at nodes i and $i+1$, the (distance, speed) pairs along the route are (x_i, s_i) and (x_{i+1}, s_{i+1}). We linearly interpolate on distance to get the speeds between the two nodes.

With speed as a function of distance, we have to solve a differential equation to get distance as a function of time, which is what we need to generate points along the route. For example, with linear interpolation along an edge, we have this relationship between speed dx/dt and distance x:

$$\frac{dx}{dt} = mx + b \tag{3}$$

With the initial condition that $x = 0$ when $t = 0$, the solution in terms of t is

$$x = \frac{1}{m}\left(e^{mt + \ln b} - b\right) \tag{4}$$

We move along the route in x as we increment t with whatever Δt we choose. For a computed x along the route, we convert to latitude/longitude using our knowledge of the lengths and coordinates of the route's constituent edges. The result of this step is a sequence of time-stamped latitude/longitude pairs along the route, sampled at whatever frequency we chose. Figure 9 shows the result of this step, where points have been filled in at one per second according to randomly chosen speeds. These points represent the locations where the simulated driver made GPS measurements.

7 GPS Noise

As a final step in simulating data from a real trip, we add noise to the simulated latitude/longitude points. This is not to obfuscate the data, but to make it look more realistic to a potential attacker. Although there are statistics published on GPS

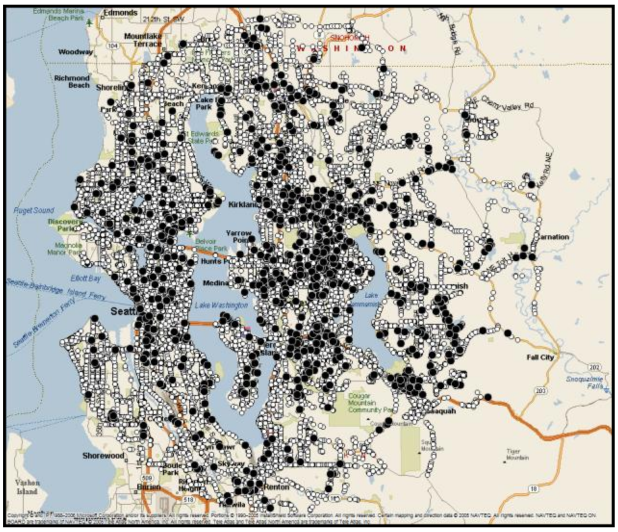


Figure 8: GPS noise varies with location. The white dots show all the points where we estimated the standard deviation of GPS noise. The black dots show the 5% of points with the largest standard deviation.

inaccuracy, *e.g.* [16], we chose to compute our own statistics from our data. In section 3, we explained how we matched each measured GPS point to a point on a nearby road. We regard the matched point as the driver's actual location, giving us differences in distance for computing statistics. Adopting the Gaussian assumption from [16], we further assume that the GPS errors have zero mean, leaving only the standard deviation as the parameter of interest. Our observations show that some GPS measurements are outliers, so we use a robust estimate of the standard deviation, the median absolute distance (MAD) [14]. The MAD gives a valid estimate of standard deviation even if up to half the values are outliers. This is why, even if up to half our GPS measurements are outliers or mismatched to a road, we can still compute a reasonable estimate of GPS standard deviation. If the GPS errors are d_i, the MAD formula is

$$\sigma = 1.4826 \cdot \text{median}|d_i - \text{median}(d_i)| \qquad (5)$$

Here, since we assume that GPS error has zero mean, we replace $\text{median}(d_i)$ with zero. The factor of 1.4826 makes the estimate consistent for Gaussian distributions. We computed $\sigma = 7.65$ meters using data from all our subjects.

Our observations also show that GPS error varies with location, with higher errors perhaps coming in areas with more obstacles to prevent a clear view of the GPS satellites. With this in mind, we compute a separate GPS error standard deviation for each node we observed in our GPS data. Specifically, for each GPS point matched to a road, we associate that error to the nearest road node and compute each node's standard deviation from its associated errors.

Figure 8 shows in black the 5% of nodes with the highest GPS error. Although there is no obvious pattern, there are several clear clusters of points, indicating areas of extended disruption, caused possibly by trees or buildings.

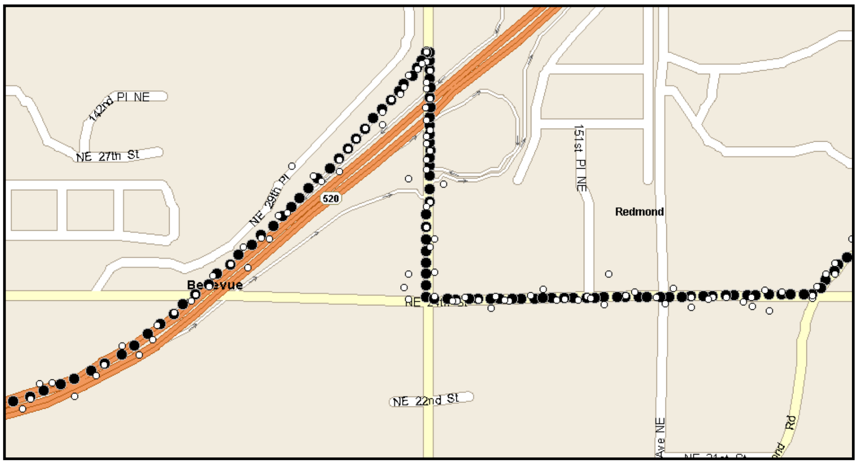


Figure 9: The larger black dots show points sampled at a rate of one per second, filled in along edges according to randomly generate speeds. The smaller, white dots have had spatially varying, GPS noise added to them.

To add realistic GPS noise to our traces, for each point, we first generate a random direction with a uniform distribution, $\theta \sim U(0,2\pi)$. We then find the σ associated with the nearest node and generate a random magnitude $d \sim N(0,\sigma)$. The point is then moved by $(\Delta x, \Delta y) = (d\cos\theta, d\sin\theta)$. Figure 9 shows a section of one of our traces, with and without added noise.

Adding noise is the last step of our process. We note that this is the only step that does not abstract away the specific training region. Our simulated start and end points, routes, and speeds are based on generic features that could be extracted from any city without taking GPS data there (*i.e.* the road network and USGS ground cover data). A simple alternative to site-specific training for GPS noise would be to use the same value of σ everywhere. A more interesting alternative would be to learn a model that infers σ as a function of relevant features, perhaps USGS ground cover and the density of nearby buildings.

8 Summary

To summarize, this is the list of steps used to generate a false trip:

1. Trip endpoints – Use features from Table 1 to compute the probability of each node serving as a trip endpoint. The start of the trip is chosen according to these probabilities. The end of the trip is chosen according to the same probabilities, augmented with the probability distribution of trip times given in Figure 5. This gives realistic starting and ending points and realistic trip times.
2. Trip speeds – Based on simple learning from GPS traces, compute probabilistic speed distributions for each node as a function of the posted

speed limits and types of road approaching and departing each node. For example, Figure 6 gives a speed distribution for going from a limited access highway with a certain speed limit to a ramp with another speed limit.

3. Random routes – Given a random start and end of a trip, generate a route. Instead of using posted speed limits to compute the minimum time route, we use speeds randomly drawn from the speed distributions in the previous step. This makes the routes somewhat random and unpredictable, but still reasonable.
4. Points along route – Draw another set of random speeds at nodes along the computed route. Linear interpolation gives the speed at any point along the route, and solving a simple differential equation gives distance along the route as a function of time.
5. GPS noise – Add spatially varying GPS noise to the previously computed points on the route. The spatial variation was computed based on our sampled GPS data.

9 Discussion

The steps outlined in the preceding sections constitute a method for generating realistic, false trips for location privacy. Some false trips generated from the method are shown in Figure 10. As a way to enhance privacy, the technique's ultimate utility comes in whether or not an attacker could distinguish the false trips from real ones. The likely attack method would be to find some characteristic of real trips and test to see which trips pass the test. The current method incorporates the major characteristics of everyday trips.

Techniques like this should be subjected to scrutiny from unbiased researchers posing as attackers. If they find an unmodeled characteristic that distinguishes false trips from true trips, that characteristic should be incorporated into the simulation. Toward this end, we have made available 1000 simulated trips and 10 real trips from our test area available on a public Web site[1]. The simulated trips come from the technique described in this paper. This site also contains a movie showing the progress of the 1000 false trips on a map. The movie shows that most trips start and end in more urban areas, with fewer in less populated regions.

While ours is one of the first efforts to produce realistic trips for location privacy, there are published criteria for trip simulation. One list of criteria comes from a survey of vehicular simulation techniques for mobile ad hoc networks [5]. Their five "macro-mobility" criteria apply to our technique:

[1] http://research.microsoft.com/en-us/um/people/jckrumm/RealisticDrivingTrips/data.htm

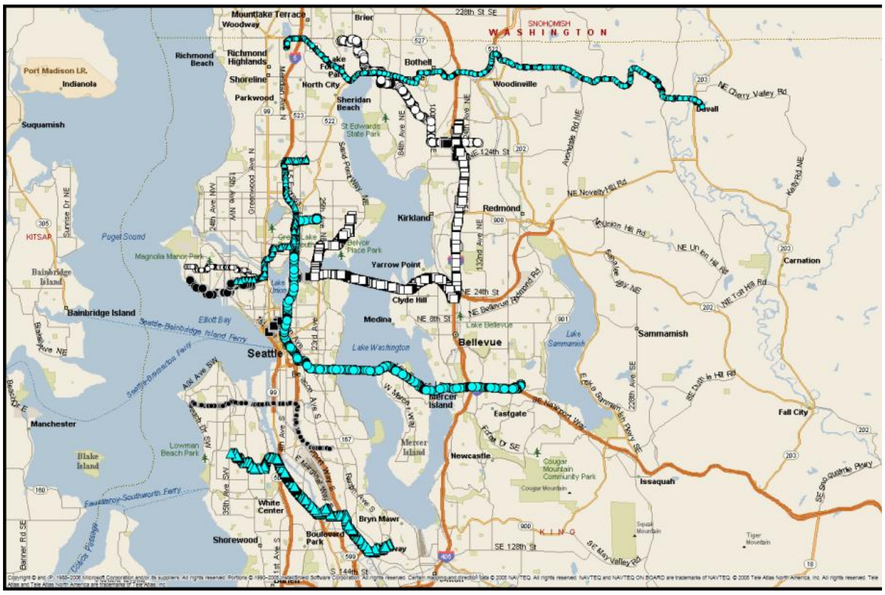


Figure 10: These are ten false trips generated by our method

- **Graph** – Vehicular models that move on a map-derived graph, like ours, are considered more realistic.
- **Initial Destination and Position** – Our endpoints are not random. They are restricted to the graph and represent characteristics of the endpoints we observe in data.
- **Trip Generation** – Endpoints can be generated based on likely activities of drivers (*e.g.* shopping, entertainment). Our models do not account for this.
- **Path Computation** – Our computed routes are based on random, but plausible, road speeds and thus demonstrate variability similar to actual drivers.
- **Velocity** – We take driving speeds from probability distributions based on our GPS data.

Another list of criteria, for a related purpose, comes from Duckham *et al.*'s [2] speculation on how a privacy attacker might attempt to refine obfuscated location data. The same refinement techniques could be applied to filter out false reports:

- **Maximum/minimum/constant Speed** – Road speeds that deviate significantly from normal are suspicious. Our trips use speeds derived from observations.
- **Connected Refinement** – An attacker would check that a sequence of location reports adheres to a connected graph of locations. Our false trips are consistent with the road network.
- **Goal-directed Refinement** – A trip that wanders aimlessly is unlikely. Our trips move toward a goal, but they do not always follow the optimal path according to published speed limits, thereby enhancing realism.

The benchmark for privacy-related, false trips is the random walk methods in Kido *et al.* [9], which is the only previous attempt we know of. Our trips are sensitive to the road network, the locations where drivers start trips, their destinations, the randomness of their routes, and the speeds they drive. While this is a significant improvement over previous work, there are more trip features to consider:

- **Time Sensitivity** – All our models disregard the time of day, day of the week, *etc*. It is likely that trip characteristics vary with time. For instance, commuters normally leave residential areas in the morning to drive to commercial areas. However, our goal is to simulate plausible trips, not aggregate traffic flows, so time sensitivity is not critically important. It would be easy to retrain our driver behavior models with different time slices.
- **Stops** – Without knowledge of the locations of stop signs, stop lights, and traffic slowdowns, we could not adequately model stops during a trip. While our speed distributions do admit very slow speeds, we do not explicitly model stops nor their durations.
- **GPS Outliers** – We know that GPS receivers occasionally produce outliers, sometimes repeatedly whenever they return to a certain place. We do not attempt to model this.

Increasing realism is not the only way to improve the effectiveness of false reports. It is also worth considering making the true report look more like a false one in order to confuse an attacker. For instance, if the false reports lack fidelity on a micro scale (*e.g.* lane selection before a turn, brief stops), it may be easier to simply add more noise to the false *and* true reports to cover minor infidelities. Decreasing precision and accuracy of location reports is an acknowledged method for protecting privacy [1], and it can make it more difficult for an attacker to distinguish real trips from false ones. Likewise, instead of adding outliers to the false reports, it may be easier to filter outliers from the real reports.

Still unresolved is *when* a privacy-minded client would report false trips – continuously, only while the client is actually moving, random times? It would be possible to build a higher level process that invokes our realistic trips at realistic times of the day to simulate movement and stop patterns over extended periods of time.

10 Conclusion

Generating false trips is one way to enhance location privacy. We generate false trips by abstracting probabilistic models from real trips and using these probabilities to generate random start and end points, random routes, random speeds, and random GPS noise.

References

1. Duckham, M. and L. Kulik, A Formal Model of Obfuscation and Negotiation for Location Privacy, in 3rd International Conference on Pervasive Computing (Pervasive 2005). 2005, Springer: Munich, Germany. p. 152-170.
2. Duckham, M., L. Kulik, and A. Birtley, A Spatiotemporal Model of Strategies and Counter Strategies for Location Privacy Protection, in 4th International Conference on Geographic Information Science (GIScience 2006). 2006, Springer: Münster, Germany. p. 47-64.
3. Gruteser, M. and D. Grunwald, Anonymous Usage of Location-Based Services Through Spatial and Temporal Cloaking, in First ACM/USENIX International Conference on Mobile Systems, Applications, and Services (MobiSys 2003). 2003, ACM Press: San Francisco, CA USA. p. 31-42.
4. Gruteser, M. and B. Hoh, On the Anonymity of Periodic Location Samples, in Second International Conference on Security in Pervasive Computing. 2005: Boppard, Germany. p. 179-192.
5. Harri, J., F. Filali, and C. Bonnet, Mobility Models for Vehicular Ad Hoc Networks: A Survey and Taxonomy. 2007, Institut Eurecom, Department of Mobile Communications: Sophia-Antipolis, FRANCE.
6. Hoh, B., et al., Enhancing Security and Privacy in Traffic-Monitoring Systems, in IEEE Pervasive Computing Magazine. 2006, IEEE. p. 38-46.
7. http://landcover.usgs.gov/ftpdownload.asp.
8. Hu, P.S. and T.R. Reuscher, Summary of Travel Trends, 2001 National Household Travel Survey. 2004, U. S. Department of Transportation, U.S. Federal Highway Administration. p. 135.
9. Kido, H., Y. Yanagisawa, and T. Satoh, An Anonymous Communication Technique Using Dummies For Location-based Services, in IEEE International Conference on Pervasive Services 2005 (ICPS2005). 2005: Santorini, Greece. p. 88-97.
10. Krumm, J., Inference Attacks on Location Tracks, in Fifth International Conference on Pervasive Computing (Pervasive 2007). 2007: Toronto, Ontario, Canada. p. 127-143.
11. Krumm, J., J. Letchner, and E. Horvitz, Map Matching with Travel Time Constraints, in Society of Automotive Engineers (SAE) 2007 World Congress. 2007: Detroit, MI USA.
12. Letchner, J., J. Krumm, and E. Horvitz, Trip Router with Individualized Preferences (TRIP): Incorporating Personalization into Route Planning, in Eighteenth Conference on Innovative Applications of Artificial Intelligence (IAAI-06). 2006: Boston, Massachusetts USA.
13. Rish, I. An Empirical Study of the Naive Bayes Classifier. in IJCAI-01 Workshop on Empirical Methods in AI. 2001.
14. Rousseeuw, P.J. and C. Croux, Alternatives to the Median Absolute Deviation. Journal of the Americal Statistical Association, 1993. **88**(424): p. 1273-1283.
15. Steffen, M., A Simple Method for Monotonic Interpolation in One Dimension. Astronomy and Astrophysics, 1990. **239**(November (II)): p. 443-450.
16. van Diggelen, F., GNSS Accuracy: Lies, Damn Lies, and Statistics, in GPS World. 2007.